Lab3

Worksheet #1: After running the cp command, what file(s) are now in your home directory? There should be at least two: a ".sh" file, and a ".py" file.

When I type in "ls –l" command, it shows:

```
total 3
drwxr-xr-x 2 changbai eecs 60 Sep 19 15:13 si618FluxSetup
```

Files shows as below:

si618FluxSetup ,si618FluxSetup/ngram-job.py and si618FluxSetup/spark-run.sh

Worksheet #2: **What is the name of the last file in the listing for HFS folder /var/si618f17?**

ataset_review.json

Worksheet #3: **What year was Einstein first mentioned (as a noun) in Google Books data?**

```
einstein_NOUN 1921   4      4
```

year:1921

Worksheet #4: **After the Spark job completes, what are three files listed in your Hadoop File System output directory ./ngrams-out?**

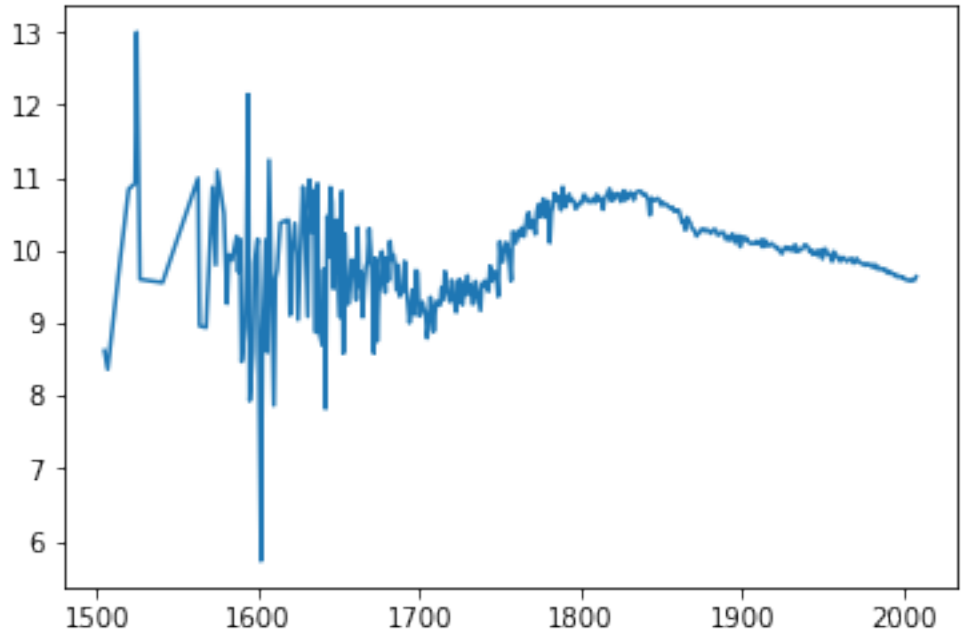
```
Found 3 items
-rw-r-----   3 changbai hadoop          0 2017-09-19 15:38 ngrams-out/_SUCCESS
-rw-r-----   3 changbai hadoop       5540 2017-09-19 15:38 ngrams-out/part-00000
-rw-r-----   3 changbai hadoop       5492 2017-09-19 15:38 ngrams-out/part-00001
```

Worksheet #5: **What were the average word lengths observed in books from the years 1520, 1597, and 1598 ?**

```
(1592, 8.937106918238994)
(1594, 12.142857142857142)
(1520, 10.84)
```

6. Bonus Challenge

Using the output file that is placed in your HFS output directory, produce a scatterplot of how average word length has changed over the full time period of the dataset (using your favorite spreadsheet or data viz program). Insert it below.


13
12
11
10
9
8
7
6
1500
1600
1700
1800
1900
2000

Lab3_Bonus

September 24, 2017

In [53]:
```
import pandas as pd
import numpy as np
```

In [12]:
```
data = pd.read_csv('ngrams-output.txt',sep = '\n',header = None)
```

In [13]:
```
data.columns = ['a']
```

In [21]:
```
data.head
```

Out[21]:
```
<bound method NDFrame.head of                                 a
0      (1564, 8.95897435897436)
1                 (1568, 8.9375)
2     (1572, 10.86864406779661)
3     (1574, 9.798076923076923)
4     (1582, 9.938983050847458)
5     (1584, 9.859756097560975)
6                   (1588, 9.7)
7     (1590, 8.466989436619718)
8     (1592, 8.937106918238994)
9    (1594, 12.142857142857142)
10                (1598, 9.832)
11   (1600, 10.153295128939828)
12   (1602, 5.7272727272727275)
13    (1606, 8.594594594594595)
14    (1610, 7.866666666666666)
15     (1612, 9.72972972972973)
16                (1614, 10.375)
17   (1618, 10.413793103448276)
18    (1620, 9.106382978723405)
19    (1624, 9.962686567164178)
20    (1626, 9.645833333333334)
21                (1628, 10.875)
22     (1630, 9.87683284457478)
23    (1632, 10.96551724137931)
24   (1634, 10.240663900414937)
25    (1636, 8.869565217391305)
26     (1638, 8.90632318501171)
27    (1640, 8.696629213483146)
```

```
28     (1642, 7.82051282051282)
29    (1644, 9.903419316136773)
..                          ...
395   (1959, 9.921816098977523)
396   (1961, 9.845579194365698)
397   (1963, 9.899037046700808)
398   (1965, 9.845548066952784)
399   (1967, 9.878510422578803)
400   (1969, 9.826507100795467)
401   (1971, 9.888930615152034)
402   (1973, 9.841120218165624)
403   (1975, 9.804956605112249)
404   (1977, 9.791682083327476)
405   (1979, 9.802351405104226)
406   (1981, 9.761339697764583)
407   (1983, 9.794734450150585)
408   (1985, 9.761445831650367)
409   (1987, 9.738816909839922)
410   (1989, 9.710144230777455)
411   (1991, 9.679409295296683)
412   (1993, 9.681129990307564)
413   (1995, 9.669703832600856)
414    (1997, 9.64522427520491)
415   (1999, 9.622537124815894)
416   (2001, 9.598801263623741)
417   (2003, 9.584914653965292)
418    (2005, 9.57700619486979)
419   (2007, 9.589189270312401)
420   (1505, 8.619047619047619)
421   (1507, 8.361702127659575)
422   (1515, 9.918067226890756)
423                (1525, 13.0)
424                 (1527, 9.6)

[425 rows x 1 columns]>
```

In [165]:
```
year = []
ave = []
num_dict = {}
```

In [166]:
```
import collections

def function(data):
    #num_list = data.loc[0].tolist()[0].replace('(','').replace(')','').split(',')
    num_list = list(data)
    for num in num_list:
        num = num.replace('(','').replace(')','').split(',')
        num = list(map(float,num))
```

```
            i = iter(num)
            num_dict.update(dict(zip(i,i)))

            # sort by year
            od = collections.OrderedDict(sorted(num_dict.items()))
        for k,v in od.items():
            year.append(k)
            ave.append(v)
```

```
In [167]: data.apply(function,axis = 0)
```

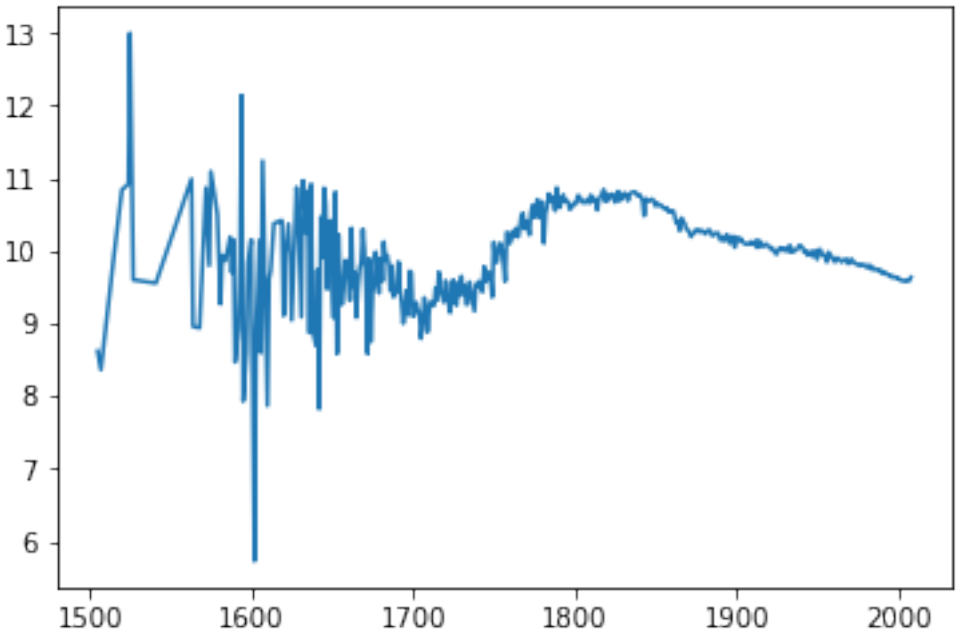
```
Out[167]: a    None
          dtype: object
```

```
In [168]: len(year)
```

```
Out[168]: 425
```

```
In [173]: import matplotlib.pyplot as plt

          plt.plot(year, ave)
          plt.figure(figsize=(20,10),dpi=80, facecolor='w', edgecolor='k')
          plt.show()
```



```
<matplotlib.figure.Figure at 0x1173833c8>
```

```
In [127]: a = ['11','12']
          a = list(map(int,a))
          i = iter(a)
          b = dict(zip(i,i))
          b

Out[127]: {11: 12}

In [138]: d = {2:3, 1:89, 4:5, 3:0}

          od = collections.OrderedDict(sorted(d.items()))

          od

Out[138]: OrderedDict([(1, 89), (2, 3), (3, 0), (4, 5)])
```